Integrating Serious Game Development into a Systems and Software Engineering Curriculum

Nestor Suat-Rojas[1]; Oscar Agudelo Varela[1]; Patricia Chávez-Ávila[2]
[1] *Faculty of Basic Sciences and Engineering, Universidad de los Llanos, Villavicencio, Colombia*
[2] *Faculty of Human Sciences and Education, Universidad de los Llanos, Villavicencio, Colombia*
nestor.suat@unillanos.edu.co, oscar.agudelo@unillanos.edu.co, pchavez@unillanos.edu.co

Abstract*– This study examines the integration of serious game development within an undergraduate systems and software engineering curriculum using a Project-Based Learning (PBL) approach. Engineering students engage in the design and implementation of a narrative-driven serious game addressing the armed conflict in Colombia as a socially grounded application context. The development activity is framed as a system-oriented engineering task, involving architectural design, modular software development, interaction modeling, system integration, and iterative refinement under realistic technical and organizational constraints. The work focuses on engineering learning processes associated with sustained engagement in the development of a complex software system. It is organized around multiple development phases, including system analysis, architectural definition, component implementation, and integration activities. These phases provide a structured context in which students engage with system decomposition, interface definition, state management, and technical trade-offs across the software development lifecycle. By embedding the development task within the formal curriculum, the project supports extended engagement with open-ended engineering problems that require coordination among interdependent system components and adaptation to evolving requirements. The socially grounded narrative context introduces additional design considerations related to representation, interaction sequencing, and system behavior, which must be addressed through architectural and software engineering decisions. This paper contributes to understanding how complex, system-level software development projects situated within undergraduate engineering education can support systems thinking and the development of core engineering competencies.

Keywords*— Project-based learning; Systems engineering curriculum; Game development-based learning; Serious games; Engineering education

I. Introduction

Engineering education worldwide is increasingly moving away from traditional lecture-based instruction toward pedagogical models that reflect the complexity, uncertainty, and interdisciplinarity of real-world engineering practice. Conventional courses often isolate technical content from the open-ended and ill-structured problems engineers encounter in professional contexts, limiting opportunities for students to engage in authentic design challenges and collaborative problem solving [1], [2]. As a result, Project-Based Learning (PBL) has become a widely adopted instructional strategy within engineering curricula. In this context, the study is situated within ongoing efforts to enhance undergraduate engineering education by embedding authentic, system-oriented development tasks within the curriculum.

PBL situates students in the active design, development, and implementation of meaningful artifacts that mirror the processes and challenges of real engineering tasks. Rather than following prescriptive instructions or focusing on isolated concepts, students engage in inquiry, decision-making, teamwork, and iterative refinement, supporting the development of both technical competencies and professional capacities such as communication, coordination, and interdisciplinary integration [2], [3]. Systematic reviews of the literature indicate that PBL is associated with deep engagement and improved learning outcomes when students work with complex problem contexts [4], [5]. Within engineering education, PBL is frequently aligned with frameworks such as CDIO (Conceive–Design–Implement–Operate), which emphasize full lifecycle engineering experiences and integration of knowledge and skills in realistic settings [5].

Within this shift toward active and authentic learning, serious game development represents a particularly rich context for engineering education. Unlike conventional project assignments, the design and implementation of a serious game requires integration of software engineering principles, system architecture, human–computer interaction, testing and validation, and iterative refinement within a single engineered product. From an educational perspective, this approach aligns with Game Development–Based Learning, a pedagogical strategy in which students develop technical and professional competencies through the construction of complete game systems. This integration reflects the multidisciplinary nature of contemporary engineering practice, in which software systems must be designed with attention to technical performance, user interaction, and contextual limitations.

In this study, the serious game under development is intended to support educational engagement with the armed conflict in Colombia, addressing themes related to historical memory, social impact, and collective understanding among secondary-level learners. Designing a game for this context introduces engineering challenges, including sensitivity to content representation, narrative structure, user diversity, and ethical considerations, all of which must be translated into system requirements, architectural decisions, and interaction mechanisms. These contextual demands extend the engineering task beyond purely technical implementation and require systematic reasoning about how software behavior, interaction design, and system structure respond to real-world social conditions.

While serious games have been examined as instructional tools within engineering education [6], existing work has largely emphasized learning outcomes associated with using such games. In contrast, this study focuses on the engineering learning processes that emerge during the development of a serious game situated in a socially and historically complex context. By treating the game as an engineered software system, this study examines how design decisions, architectural modularization, system integration, and iterative refinement are shaped by both technical and contextual constraints.

Accordingly, this study presents the instructional and methodological framework supporting serious game development within undergraduate engineering education. By focusing on system-oriented development, this work contributes to discussions on how socially grounded software engineering tasks can support engineering learning within project-based educational settings.

II. Educational Context

The project is situated within the systems engineering undergraduate program at the Universidad de los Llanos (Unillanos), a public university in Colombia. The initiative is integrated into the students' formal academic trajectory and is aligned with degree requirements within the engineering curriculum. Participation serves as the basis for undergraduate thesis work (trabajo de grado) required for graduation.

A total of five undergraduate engineering students participate in the project. These students are involved in the design and development of the serious game as part of their final academic year, under the supervision of two faculty members from the software area of the systems engineering program. As a result, the project functions simultaneously as a pedagogical experience and as an applied engineering development effort.

Students are organized into a small development team and assume roles commonly found in professional software engineering environments, including system architecture design, gameplay logic implementation, user interface development, testing, and technical documentation. Role assignments are flexible and evolve throughout the project, allowing students to experience multiple aspects of the software development lifecycle and to negotiate responsibilities collaboratively, reflecting authentic engineering practice.

Faculty professors act as project mentors rather than directive instructors, providing technical guidance, feedback on design decisions, and oversight of development milestones while preserving student autonomy. This mentoring model encourages independent problem solving and supports reflective engagement with engineering decisions made during the project.

The development task is intentionally structured under realistic conditions, including limited development time, fixed hardware and software resources, evolving functional requirements, and coordination with non-technical stakeholders external to the engineering program. Through exposure to such conditions, students encounter challenges typical of real-world software engineering projects, such as requirement ambiguity, integration issues, and resource limitations, which contribute directly to their engineering learning experience.

III. Serious Games as Engineered Systems

A. Engineering System Perspective

From an educational standpoint, the serious game developed in this project is treated as an engineered system rather than as a pedagogical artifact or instructional medium. The game is conceptualized as a software system composed of multiple interdependent subsystems, including a game engine, narrative logic modules, interaction mechanics, user interface components, and data handling elements. Each subsystem introduces specific functional requirements, challenges, and performance considerations that must be addressed through structured engineering design and implementation processes.

This systems perspective aligns with established views in software engineering, where complex applications are understood as compositions of interacting components whose behavior emerges from integration rather than from isolated modules [7]. Students are therefore required not only to implement individual features, but also to reason about system coherence, interface compatibility, and robustness under changing requirements. Testing and debugging activities are framed as integral engineering tasks necessary to validate system behavior and ensure functional stability across development iterations.

Game engine selection constitutes a critical engineering decision within the project. Rather than selecting an engine based on familiarity or aesthetic affordances, students evaluate candidate engines according to engineering-relevant criteria, including computational performance, scalability, development complexity, cross-platform support, documentation quality, and availability of reusable libraries. These considerations mirror professional software selection processes, in which engineers must balance technical capabilities against tight resource budgets and long-term maintainability [8].

The serious game context further amplifies engineering complexity by requiring the coordination of real-time interaction, event-driven logic, and user interface

responsiveness within a single system. Such requirements expose students to challenges commonly encountered in interactive software systems, including synchronization, state management, and performance optimization. Prior research on serious game development emphasizes that these challenges necessitate disciplined engineering practices and structured development workflows comparable to those used in industrial software projects [9].

By framing serious game development as a systems engineering problem, the project enables students to engage with core engineering concepts such as modularity, abstraction, trade-off analysis, and lifecycle thinking. This approach reinforces the view that serious games are not merely educational tools, but complex engineered systems whose development provides a meaningful context for engineering learning [10].

B. Software Architecture and Narrative Modularization

Narrative elements within the game are modularized to support flexible system behavior and future scalability. Students implement narrative components as discrete modules governed by state transitions and event triggers, allowing the system to manage user progression without hard-coded linear paths. This approach aligns with established software modularity principles, where complex systems are decomposed into cohesive units that can be developed, tested, and evolved with reduced coupling and clearer interfaces [7], [8], [11].

Modularization promotes separation of concerns, reusability, and maintainability, all of which are critical architectural attributes in robust software systems [7], [11]. In the context of narrative logic, each story module encapsulates its own state, transitions, and event responses, allowing multiple narrative threads to operate and interact without being tightly coupled within a single monolithic logic structure. Such decomposition mitigates coupling between narrative sequences and supports both isolated component testing and integration testing as the system evolves.

By framing narrative implementation as a software architecture problem, students confront challenges related to state management, data persistence, context propagation, and extensibility. These challenges mirror architectural concerns in larger software systems beyond gaming, where event notification, decoupled components, and state-transition logic are fundamental to system correctness and evolution [7], [8], [12]. For instance, event-driven interaction styles—often realized through publish/subscribe communication—support decoupling between system components and facilitate modular reconfiguration as requirements change [12].

C. Interaction Mechanics

Interaction mechanics are treated as system-level behaviors rather than purely creative design elements. Students define input–response mappings, feedback loops, and interaction constraints that govern user–system engagement. Implementing these mechanics requires careful consideration of timing, responsiveness, error handling, and performance optimization, particularly in interactive systems where delays or inconsistencies directly affect perceived system quality and usability [13].

Through this process, students gain experience in designing interactive systems in which user actions trigger state transitions and event propagation across system components. This reinforces core engineering concepts related to control flow, event-driven programming, and user-centered system design, highlighting the close relationship between interaction design decisions and underlying software architecture [14]. By treating interaction mechanics as engineered behaviors, students are encouraged to reason about system responsiveness and reliability using formal engineering principles.

IV. Methodological Framework

The study adopts a PBL framework structured to support systematic examination of engineering learning processes within an authentic software development context. Undergraduate engineering students engage in the design and implementation of a serious game as a sustained development task, requiring iterative design, integration, and refinement across multiple development cycles.

The methodological design positions the development activity as an observational setting in which engineering competencies emerge through interaction with technical constraints, system requirements, and collaborative workflows. The study spans the full software development lifecycle, including architectural design, component implementation, system integration, testing, and documentation, enabling analysis of engineering practices as they occur during real development work.

Learning objectives are formulated as engineering competencies that guide both instructional design and data collection. These competencies include system design and architectural planning, modular software implementation, collaborative development supported by version control tools, debugging and performance evaluation, and technical documentation. Each competency is addressed through development tasks embedded within the study timeline.

Evaluation emphasizes process-level evidence of engineering activity. Data sources include software artifacts such as architectural diagrams, source code repositories, version histories, test reports, and technical documentation generated throughout the study. In addition, structured student reflections are collected at predefined milestones to capture decision rationales, constraint management strategies, and responses to technical challenges encountered during development and integration.

Professor supervision is organized around periodic technical reviews and milestone evaluations. Professors also provide targeted feedback on architectural coherence, implementation quality, and system behavior, while maintaining student responsibility for design decisions. Guided reflection sessions support articulation of engineering reasoning and contribute to documentation of how technical

choices align with system requirements and development constraints.

This methodological framework enables examination of engineering learning as a function of sustained engagement with a complex software system, providing a structured basis for analyzing how project-based development activities support the acquisition and application of core engineering competencies.

V. Expected Engineering Outcome

The primary expected outcome of the project is a functional prototype of a single-player, narrative-driven serious game developed as a modular software system. The prototype provides a concrete setting in which architectural design, interaction logic, and system integration can be explored through sustained software development activity conducted under realistic constraints.

The game is structured around scenario-based interaction, where players engage with a sequence of narrative situations related to the Colombian armed conflict through explicit decision points. Player input is handled through choice-based interactions that update internal narrative states, shaping progression across scenarios and determining subsequent system responses.

At the architectural level, the system is organized around a modular narrative engine composed of discrete story units. Each unit defines its own narrative context, interaction options, and progression conditions. Narrative flow is coordinated through event-driven control logic, allowing player decisions to influence subsequent scenarios while maintaining clear separation between narrative content, interaction handling, and core system coordination.

Interaction mechanics are defined through rule-based mappings between player actions and system responses. Player choices generate events that propagate across system components, updating narrative state variables and triggering associated feedback mechanisms. The user interface mediates this interaction by presenting choices, conveying system feedback, and maintaining consistent timing and responsiveness throughout gameplay.

The prototype is designed to accommodate incremental extension and iterative modification, enabling additional narrative scenarios to be incorporated without restructuring the underlying system. This approach supports maintainability, scalability, and clarity of system behavior, allowing engineering decisions related to modularity, integration, and state management to be examined as part of the development process.

The game also operates as a socially grounded interactive system in which technical design choices are inseparable from contextual meaning. Narrative scenarios related to the Colombian armed conflict frame system behavior in ways that foreground issues of representation, sequencing, and consequence, requiring developers to account for how architectural structure, interaction timing, and state transitions influence user interpretation and engagement in sensitive real-world contexts.

VI. Conclusion

This work presents an ongoing study of serious game development as a system-oriented engineering task within undergraduate engineering education. By treating the game as a modular software system, the project foregrounds core engineering activities such as architectural design, system integration, interaction modeling, and iterative refinement, while situating development within a socially meaningful application context. Implemented within a PBL framework, the study provides a sustained development setting in which students engage with complex, open-ended engineering problems under realistic constraints.

Subsequent phases of the study will focus on systematic examination of the engineering materials produced throughout development, including architectural designs, source code evolution, integration records, and technical documentation. Analysis of these materials will support investigation of architectural reasoning, collaboration practices, and technical decision-making across development stages. Future iterations may also explore variations in system architecture, development tools, or project scope to assess their influence on observed engineering processes.

By positioning serious game development as a system-level engineering activity, this work contributes to ongoing efforts to understand how authentic software development contexts support engineering learning. The study aims to inform the design of systems and software engineering curricula that emphasize system thinking, integration, and technical decision-making in conditions that reflect professional engineering practice.

Acknowledgment

The authors acknowledge the support of Universidad de los Llanos for funding the research project (C09-02-2026-016), approved under Act No. 31 on November 26, 2025, which provided the academic context within which this review was developed.

References

[1] I. de los Ríos, A. Cazorla, J. M. Díaz-Puente, and J. L. Yagüe, "Project-based learning in engineering higher education: Two decades of teaching competences in real environments," *Procedia – Social and Behavioral Sciences*, vol. 2, no. 2, pp. 1368–1378, 2010, ISSN: 1877-0428.

[2] V. Sukackė, A. O. P. de C. Guerra, D. Ellinger, V. Carlos, S. Petronienė, L. Gaižiūnienė, S. Blanch, A. Marbà-Tallada, and A. Brose, "Towards active evidence-based learning in engineering education: A systematic literature review of PBL, PjBL, and CBL," *Sustainability*, vol. 14, no. 21, Art. no. 13955, 2022.

[3] M. Ramírez de Dampierre, M. C. Gaya-López, and P. J. Lara-Bercial, "Evaluation of the implementation of project-based learning in engineering programs: A review of the literature," *Education Sciences*, vol. 14, no. 10, Art. no. 1107, 2024.

[4] A. C. B. Reis, S. C. M. Barbalho, and A. C. D. Zanette, "A bibliometric and classification study of project-based learning in engineering education," *Production*, vol. 27, special issue, Art. no. e20162258, 2017.

[5] A. Shekar, "Project-based learning in engineering design education: Sharing best practices," in *Proc. 2014 ASEE Annual Conf. & Exposition*, Indianapolis, IN, USA, Jun. 2014, pp. 24-1016.

[6] M. Urgo, W. Terkaj, M. Mondellini, and G. Colombo, "Design of serious games in engineering education: An application to the configuration and analysis of manufacturing systems," *CIRP Journal of Manufacturing Science and Technology*, vol. 36, pp. 172–184, 2022, ISSN: 1755-5817.

[7] I. Sommerville, *Software Engineering*, Global ed. Boston, MA, USA: Pearson, 2016.

[8] R. S. Pressman, *Software Engineering: A Practitioner's Approach*. New York, NY, USA: Palgrave Macmillan, 2005.

[9] R. Dörner, S. Göbel, W. Effelsberg, and J. Wiemeyer, *Serious Games*. Cham, Switzerland: Springer, 2016.

[10] J. Novak, *Game Development Essentials: An Introduction*. Santa Monica, CA, USA: Novy Publishing, 2022.

[11] M. S. Thaiya, K. Julia, and S. Mbugua, "On software modular architecture: Concepts, metrics and trends," *International Journal of Computer and Organization Trends*, vol. 12, no. 1, pp. 3–10, 2022.

[12] L. Lazzari and K. Farias, "Event-driven architecture and REST architectural style: An exploratory study on modularity," *Journal of Applied Research and Technology*, vol. 21, no. 3, pp. 338–351, 2023.

[13] B. Myers, S. E. Hudson, and R. Pausch, "Past, present, and future of user interface software tools," *ACM Transactions on Computer-Human Interaction*, vol. 7, no. 1, pp. 3–28, Mar. 2000, ISSN: 1073-0516.

[14] M. Beaudouin-Lafon, "Designing interaction, not interfaces," in *Proc. Working Conf. Advanced Visual Interfaces (AVI)*, Gallipoli, Italy, May 2004, pp. 15–22.